TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Webový portál pro výuku teoretické informatiky

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 1802R022 – Informatika a logistika

Autor práce: **Ondřej Jodas**
Vedoucí práce: Ing. Lenka Kosková - Třísková

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Web portal for teaching theoretical computer science

Bachelor thesis

Study programme: B2612 – Electrical Engineering and Informatics
Study branch: 1802R022 – Informatics and Logistics

Author: **Ondřej Jodas**
Supervisor: Ing. Lenka Kosková - Třísková

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ondřej Jodas**
Osobní číslo:	**M12000288**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Informatika a logistika**
Název tématu:	**Webový portál pro výuku teoretické informatiky**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Seznamte se s existujícími implementacemi úloh (převod konečného automatu na regulární výraz, simulace gramatiky, simulace konečného automatu) a použitými technologiemi (PHP, JavaScript).
2. Navrhněte koncept portálu včetně seznamu očekávaných funkcí a návrhu architektury.
3. Proveďte rešerši existujících redakčních systémů.
4. Navrhněte administrační rozhraní portálu, buď s pomocí existujícího redakčního systému, nebo jako vlastní aplikaci.
5. Navrhněte API pro vkládání nových simulací.
6. Doplňte vlastní aplikaci s využitím navrženého API, která simuluje chování konečných deterministických i nedeterministických automatů a automatů s prázdnými přechody. Aplikace umí ukládat a načítat konfiguraci automatu z a do XML. Aplikace umí krokovat simulaci a kdykoliv ji zastavit. Aplikace umožňuje automat nakreslit i definovat tabulkou.
7. Navrhněte, jak do portálu integrovat již existující simulace.
8. Připravte dokumentaci pro budoucí uživatele autory simulací jednotlivých typů strojů i uživatele testující konkrétní stroje.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **30 - 40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] CHYTIL, M.: Automaty a gramatiky, SNTL Praha 1984
[2] HOPCROFT, John E., MOTWANI R., ULLMAN J.: Automata theory, languages and computation, Addison Wesley 2007, ISBN 0-321-45536-3

Vedoucí bakalářské práce: **Ing. Lenka Kosková - Třísková**
Ústav nových technologií a aplikované informatiky

Datum zadání bakalářské práce: **20. října 2015**

Termín odevzdání bakalářské práce: **16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

V Liberci dne 20. října 2015

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Rád bych poděkoval Ing. Lence Koskové-Třískové za její pomoc a ochotu během vedení bakalářské práce a pomoc při zpracování. Dále bych chtěl poděkovat rodině a přítelkyni za trpělivost a podporu při studiu na vysoké škole.

Abstrakt

Cílem mé bakalářské práce je vytvoření interaktivní internetové stránky. Tyto stránky budou sloužit jako pomůcka pro vyučování předmětu teoretické informatiky. V první části práce nalezne čtenář porovnání existujících redakčních systémů. Na jeho základě si vybere vhodný pro své potřeby. V následující části je popsán API a jeho úprava pro vytváření vlastního typu automatu. Poslední část je zaměřena na vytvoření a simulaci konečného automatu v aplikaci.

Klíčová slova

Konečný automat – Drupal – PHP – MySQL – Javascript – jQuery - XML

Abstract

The goal of my thesis is to create an interactive website. These sites will serve as a tool for teaching the subject of the Theoretical computer science. In the first part the reader will find a comparsion of an existing content management systems. Based of which he will choose the suitable one to fits his needs. The following section describes the API and its adaptation for creating your own automat type. The last part is focused on the creation and simulation of the final automat in aplication.

Keywords

State machine – Drupal – PHP – MySQL – Javascript – jQuery - XML

Obsah

Seznam zkratek

HTML	hypertextový značkovací jazyk pro tvorbu internetových stránek
PHP	Skriptovací programovací jazyk, který umožňuje tvorbu dynamického webu
JavaScript	klientský skriptovací jazyk
jQuery	javascriptová knihovna, která klade důraz na interakci mezi javascriptem a HTML
API	rozhraní pro programování aplikací (Application Programming Interface)
Canvas	Slouží k dynamickému vykreslování bitmap

Seznam obrázků

Seznam tabulek

1. Úvod

Fakulta Mechatroniky Technické univerzity v Liberci, kde studuji, zahrnuje předmět teoretické informatiky. Obsahem tohoto předmětu je mimo jiné seznámení s konečnými automaty a také s formálními jazyky. Právě tyto automaty a formální jazyky se staly námětem pro mojí bakalářskou práci. Toto téma jsem zkoumal po stránce teoretické a praktické.

Teoretická část mé bakalářské práce je rozdělena do čtyř hlavních kapitol. První kapitola je zaměřena na definici konečných automatů. Po této kapitole následuje rešerše existujících redakčních systémů, která nám poskytne vhodný výběr redakčního systému pro potřeby. Následující kapitola se zabývá popisem implementace konečných automatů a API pro zadávání typů konečných automatů. V poslední kapitole je popsán způsob práce s webovými stránkami, které umožní studentovi vytvořit konečný automat.

Praktická část představovala vytvoření interaktivního webového portálu. Mnou vytvořený portál umožní studentům informačních technologií vytvářet a testovat chování konečných automatů. Studentům bude sloužit jako podpora pro cvičení, kde budou řešit určité typy příkladů. Například pomocí kreslícího plátna mohou vytvořit již nadefinované typy konečných automatů. Následně jim portál umožní spuštění simulace.

2. Konečný automat

Konečným automatem obecně rozumíme systém neboli model systému, který může nabývat konečně mnoho stavů. Aktuální stav se mění na základě vnějšího podnětu. Možných podnětů není „příliš mnoho“. Platí, že pro daný stav a daný podnět je jednoznačně určeno, jaký stav bude následující, tzn. do jakého stavu systém přejde. Konkrétní konečný automat se často zadává diagramem, který také nazýváme stavový diagram či graf automatu. Jiná možnost je zadávání tabulkou, která je vhodnější pro počítačové zpracování. Tabulka je pro složitější automaty přehlednější.(1)

2.1. Základní pojmy

V této části kapitoly se čtenář seznámí s pojmy, které jsou nutné pro popsání výsledné aplikace.

2.1.1. Abeceda

Abecedou myslíme libovolnou konečnou množinu; často ji označujeme Σ. Prvky abecedy nazýváme symboly či písmena, znaky apod. Například abeceda Σ = {a, b} obsahuje dvě písmena.

Slovem, neboli řetězcem, nad abecedou Σ (též říkáme: v abecedě Σ) rozumíme libovolnou konečnou posloupnost prvků množiny Σ. Pro Σ = {a, b} je to například a,b,b,a,b ; pokud nemůže dojít k nedorozumění, píšeme takovou posloupnost obvykle bez čárek, jako abbab. Prázdné slovo „ “ je také slovem a značí se ε. (1)

2.1.2. Formální jazyk

Formální jazyk, stručně jazyk nad abecedou Σ je libovolná množina slov v abecedě Σ, tedy libovolná podmnožina Σ*.

2.1.3. Konečný deterministický automat

Konečný deterministický automat (zkráceně KDA) je dán uspořádanou pěticí $A = (Q, \Sigma, \delta, q0, F)$, kde

- Q je konečná neprázdná množina stavů,
- Σ je konečná neprázdná množina zvaná (vstupní) abeceda,
- $\delta : Q \times \Sigma \rightarrow Q$ je přechodová funkce,
- $q0 \in Q$ je počáteční (iniciální) stav a
- $F \subseteq Q$ je množina přijímajících (koncových) stavů.

Význam zápisu $\delta(q, a)$ (kde $q \in Q$, $a \in \Sigma$) je jasný. Ujasnili jsme si, že nám známá tabulka či graf se vlastně matematicky dá chápat jako funkce, která danému stavu („úkolu“) q a znaku a přiřadí následující stav. Definičním oborem funkce δ je tedy množina $\{(q, a) \mid q \in Q, a \in \Sigma\}$, tj. kartézský součin $Q \times \Sigma$. Oborem hodnot je Q. Počáteční stav je jen jeden, ale naopak přijímajících stavů může být více. (1)

2.1.4. Konečný nedeterministický automat

Nedeterministický konečný automat, zkráceně NKA, je dán uspořádanou pěticí $A = (Q, \Sigma, \delta, I, F)$, kde

- Q je konečná neprázdná množina stavů,
- Σ je konečná neprázdná abeceda,
- $\delta : Q \times \Sigma \rightarrow P(Q)$ je přechodová funkce,
- $I \subseteq Q$ je množina počátečních (iniciálních) stavů a
- $F \subseteq Q$ je množina přijímajících (koncových) stavů.

Rozdíl proti definici KDA je především v tom, že $\delta(q, a)$ teď nepředstavuje jeden stav, ale množinu stavů (z nichž může být jakýkoli vybrán jako následující). Tato množina může být i prázdná. V tomto případě výpočet nemůže pokračovat. (1)

2.1.5. Přijímání jazyka

Jazykem rozpoznávaným (přijímaným) automatem A rozumíme jazyk

$$L(A) = \{w \in \Sigma^* \mid \text{slovo } w \text{ je přijímáno } A\} = \{w \in \Sigma^* \mid q_0 \xrightarrow{w} F\}.$$

2.2. Funkce softwaru

Výsledná interaktivní webová aplikace umožní studentovi použít dané funkce v následujícím pořadí:

1. Výběr požadovaného typu automatu
2. Návrh zvoleného automatu pomocí grafického plátna
3. Uložit vytvořený automat nebo exportovat ve formátu XML k pozdějšímu pokračování v návrhu
4. Simulace vytvořených automatů

Pro vytvoření takovéto aplikace použije jeden z již existujících redakčních systémů, který umožňuje základní věci. Například připojení k databázi nebo správu webu přes administrační rozhraní a jiné další potřebné funkce pro práci s automaty. Pro vybraný redakční systém vytvoří rozšíření, které implementuje práci s konečnými automaty.

3. Výběr použitého redakčního systému

Tato kapitola je věnována existujícím řešením redakčních systému (CMS). Jejich výběr je založen na studii W3Techs[CITATION 1 \y \t \l 1029], která srovnává CMS podle počtu instalací. Studie probíhala na deseti milionech nejpoužívanějších webových adresách. Ty jsou poskytovány žebříčkem Alexa [CITATION 2 \y \t \l 1029] od společnosti Amazon.com.

Mezi tyto adresy patří:

- WordPress
- Joomla
- Drupal
- Magento
- Blogger

3.1. Ideální redakční systém

Pro srovnání a vyhodnocení ideálního redakčního systému pro naše potřeby jsem vytvořil hodnocení jednotlivých částí CMS. Každou část vyhodnotím samostatně a následně provedu srovnání. Ideální redakční systém by měl mít sto bodů, což je maximální počet bodů v hodnocení každé jednotlivé části.

1. Možnost rozšiřitelnost pluginy

   Tento bod je pro mě nejdůležitější, protože budu vytvářet pro vybraný redakční systém rozšíření. Z tohoto důvodu volím třicet bodů.

2. Postaven na jazyce PHP, javascript a MySQL

   Kombinaci těchto technologií jsem zvolil, protože je ovládám nejlépe ze všech dostupných a vedoucí bakalářské práce souhlasila. PHP a MySQL poslouží na serverové straně a javascript straně klientské pro vykreslování na grafické plátno. Použité technologie nejsou tak důležité jako bod 1. proto volím dvacet bodů.

3. Technická podpora pro vývoj

   Z důvodu, že jsem ani s jedním redakčním systémem nepracoval, je pro mě důležité, aby vybraný systém měl technickou podporu. Ta by měla být například ve formě manuálu. Opět tento bod není tak důležitý jako bod 1. a proto volím dvacet bodů.

4. Bezpečnost webu

   Pod pojmem bezpečnost si představuji, aby byl systém průběžně aktualizovaný a bezpečnostní díry opravovány. Z důvodu, že je redakční systém veřejně dostupný, je potřeba, aby byl chráněný před cross-site scripting a SQL injection. Protože není tak důležitý jako rozšiřitelnost pluginy, tak volím dvacet bodů.

5. Jednoduchost administračního rozhraní

Pro snadnou a rychlou orientaci uživatele v administračním rozhraní. Tento typ pro hodnocení je zřejmě nejméně důležitý a volím tedy deset bodů.

3.2. Srovnání redakčních systémů

V této kapitole si porovnáme prvních pět redakčních systémů a vyberu jeden, který bude nejvíce vyhovovat mým potřebám a na něm postavím aplikaci pro tvorbu a správu konečných automatů.

3.2.1. WordPress

WordPress umožňuje rozšíření pomocí pluginů (30 bodů). Je založený na technologiích PHP, MySQL, Javascript (20 bodů). Má rozsáhlou technickou dokumentaci, popřípadě uživatelské fórum s návody (20 bodů). Worpress má základní zabezpečení proti SQL injection a cross-site scripting (20 bodů). Pokud by uživatel trval na lepším zabezpečení, existuje plugin „Better WP Security“. Ten například omezuje počet chybných přihlášení do administračního rozhraní, pravidelně zálohuje databázi, přidává google reCaptcha k registraci nebo přihlášení aj. Administrační rozhraní je pro jednoduchou aplikaci zbytečně složité (5 bodů).

3.2.2. Joomla

Joomla umožňuje rozšiřitelnost pluginy (30 bodů). Je postaven na technologiích PHP, MySQL a javascript (20 bodů). Jako technická dokumentace slouží uživatelská komunita a dokumentace (20 bodů). Opět je tento systém chráněný proti cross-site scripting a sql injection (20 bodů). Administrační rozhraní je pro jednoduchou aplikaci nepřiměřeně složité (5 bodů).

3.2.3. Drupal

Drupal umožňuje rozšířit systém pomocí pluginů (30 bodů). Je založený na technologiích PHP, MySQL, Javascript (20 bodů). Má dobře propracovanou dokumentaci pro správu webových stránek i pro tvorbu vlastních pluginů (20 bodů). Drupal má ošetřené vstupy proti sql injection a cross-site scripting. Je aktualizovaný s chybovými záplatami (20 bodů). Protože se jedná o větší redakční systém, tak se neobejde bez složitějšího nastavení a složitějšího administrační rozhraní (5b).

3.2.4. Magento

Tento redakční systém je zaměřený spíše na internetové obchody. Proto nevyhovuje našim potřebám.

3.2.5. Blogger

Blogger neumožňuje vytváření vlastních pluginů. Dostupné jsou pouze některé předdefinované widgety (0 bodů). Je založený na programovacím jazyku python (0 odů). Díky předchozím dvěma bodům tento redakční systém není vhodný pro vytvoření webové aplikace.

3.3. Vyhodnocení rešerše

Název CMS	Rozšiřitelnost pluginy	Technologie	Podpora	Bezpečnost	Jednoduchost	Celkem bodů
WordPress	30	20	20	20	5	95
Joomla	30	20	20	20	5	95
Drupal	30	20	20	20	5	95
Magento	Redakční systém nevyhovuje					
Blogger	Redakční systém nevyhovuje					

Tabulka 1: Vyhodnocení rešerše

3.4. Výběr redakčního systému

Většina zkoumaných redakčních systémů jsou dnes na vysoké úrovni. Z porovnání mi vyšli tři vhodní kandidáti pro vytvoření pluginu do mého projekt. Jsou jimi WordPress, Joomla a Drupal. Všechny tři systémy lze rozšířit o vlastní plugin a jsou postavené na technologiích PHP, MySQL a javaScript. Mají propracovanou technickou dokumentaci, a proto by vytvoření vlastního pluginu nemělo být složité. Podle vlastních preferencí jsem vybral redakční systém Drupal, se kterým vedoucí práce souhlasila.

4. Implementace konečných automatů

4.1. Adresářová struktura webových stránek

V této kapitole přiblížím rozvržení a umístění souborů na serveru. V kořenovém adresáři je umístěno jádro Drupalu a aplikace je vytvořena jako modul pro tento redakční systém. Modul je v mém případě v Drupalu umístěn ve složce /modules/custom/TheoreticalComputerScience/. Dále si popíši, jak jsou tedy soubory umístěny v tomto modulu.

/src/

Základní složka modulu webových stránek, kde jsou umístěny soubory pro routování url adres, informace o modulu. Tyto informace představují název, popis a použitou verze Drupalu.

/src/Controller

Tato složka slouží pro umístění souborů, které pracují s programem

/src/css/

Zde jsou umístěné kaskádové styly na úpravu vzhledu stránek

/src/Form/

Složka pro všechny formuláře použité na webových stránkách

/src/images/

Složka pro umístění obrázků

/src/js/

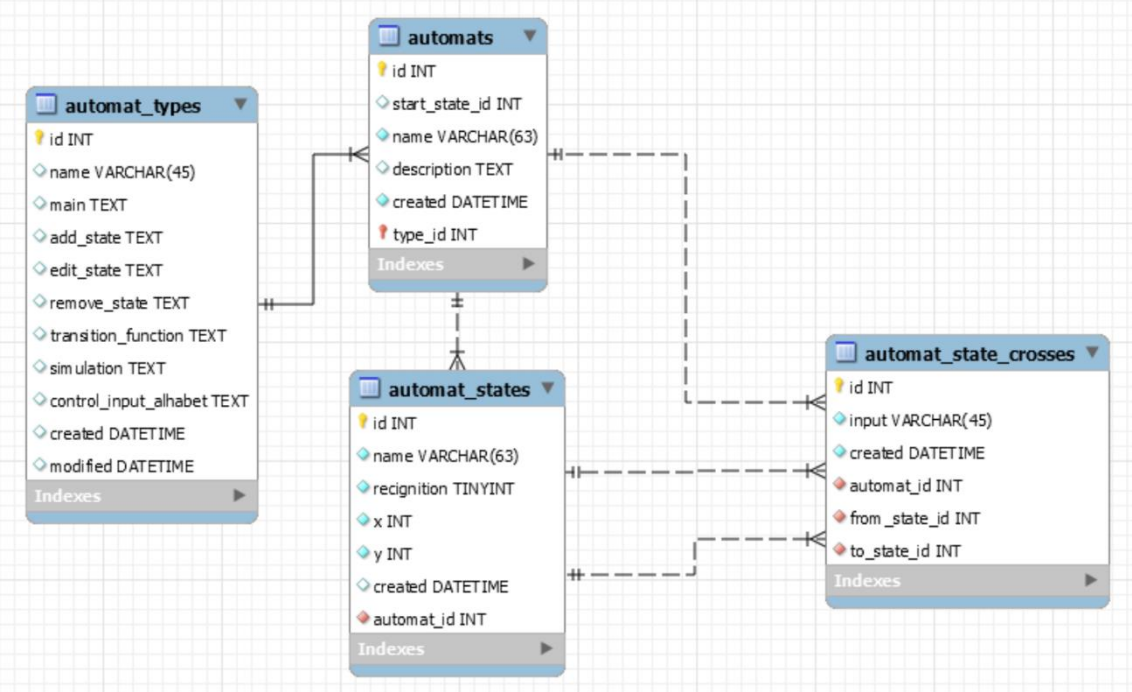
Zde jsou umístěny všechny javascriptové soubory včetně API

4.2. Databázová struktura

Obrázek 1 v této podkapitole představuje uložení automatu i jeho součástí v dané databázi. Aplikace má samozřejmě více databázových tabulek, ale ty jsou součástí standartní distribuce Drupalu. Samotný modul pro tvorbu konečných automatů se skládá z níže uvedených tabulek:

- Automat_types – zde se ukládají všechny typy automatů, ze kterých se načítá dynamická část API (popsáno v kapitole 4.4.)

- Automats – tabulka pro uložení a přehled automatů. Zde ukládám pouze název, popis a počáteční id stavu, ze kterého automat začíná
- Automat_states – stavy automatů a jejich parametry jako jsou jejich názvy, pozice, zda se jedná o stav rozpoznávací a samozřejmě i k jakému patří automatu
- Automat_state_crosses – přechodové funkce mezi automaty. Zde jsem vytvořil dvojitou 1:N vazbu na tabulky automat_states z toho důvodu, že potřebuji znát stavy počáteční a koncový. Dále tabulka obsahuje hodnotu přechodu.



Obrázek 1: Databázová struktura aplikace

4.3. Systémové komponenty

4.3.1. Routování

Jedna z nezbytných součástí moderních webových stránek je routování. Routování překládá URL adresu na akci presenteru a naopak.

V Drupalu je pro routování vyhrazen specifický soubor, který se v daném případě nazývá TheoreticalComputerScience.routing.yml. V tomto souboru jsou nadefinované veškeré routy, které se používají v mém projektu.

```
TheoreticalComputerScience.content:
  path: '/automat'
  cache: false
  defaults:
    _controller: '\Drupal\TheoreticalComputerScience\Controller\StateMachineController::content'
  requirements:
    _permission: 'access content'
```

Obrázek 2: Routování

Na obrázku 2 je vidět, že URL adresa /automat zavolá presenter třídu StateMachineController a metodu content(). Přístup na tuto adresu mají všichni návštěvníci stránek bez omezení, tedy bez přihlášení. Dále tato stránka nebude využívat cache.

4.3.2. Knihovny JS a CSS

Knihovny jsou další využívanou částí v projektu. Tyto knihovny mohou nadefinovat skupinu JS a CSS, které budou využity na určitých stránkách.

```
grafical_add:
  version: "1.x"
  header: true
  css:
      theme:
        src/css/addAutomatGrafical.css: {}
  js:
      src/js/automat.js: {}
      src/js/addAutomatGrafical.js: {}
  dependencies:
    - core/jquery
    - core/jquery.ui
    - core/jquery.ui.dialog
    - core/jquery.ui.widget
```

Obrázek 3: Knihovny JS a CSS

Z obrázku 3 je patrné, že vytvoříme skupinu grafical_add, která využívá CSS addAutomatGrafical.css a javascript automat.js a addAutomatGraficcal.js. Protože je v knihovnách javascriptu používáno jQuery, vytvoříme podmíněnou závislost pomocí dependencies. Načtení provedeme v kódu tak, jak máme možnost vidět na obrázku 4.

```php
$build = array(
    '#type' => 'markup',
    '#markup' => $output,
    '#attached' => array(
        'library' => array(
            'TheoreticalComputerScience/grafical_add',
        )
    )
);
```

Obrázek 4: Načtení JS a CSS knihovny

4.4. Popis API

V této kapitole si popíšeme fungování a strukturu javascriptového API, které slouží pro vytváření a simulaci konečných automatů

4.4.1. Struktura

API je postaveno jako jQuery widget neboli ovládací prvek. Základní část je pojmenována jako automat.base. Zde jsou umístěny všechny funkce, které jsou společné jak pro vytváření, tak pro simulaci konečných automatů. Mezi hlavní funkce patří zejména tyto:

- _drawCircle() – vytvoří javasriptový objekt typu StateObject, neboli stav a vykreslí ho na kreslící plátno
- _drawLine() – nakreslí na zadaný canvas přechod mezi dvěma stavy
- _drawArrow() – nakreslí na kreslící plátno šipku ke konečnému stavu
- _drawLineText() – na kreslící plátno vypíše text k zadanému přechodu
- _getStateIndex() – tato funkce vrátí klíč pod kterým je uložený stav v poli
- _selectState() – podle souřadnic x a y (například po kliknutí myší) vrátí funkce vybraný stav
- _refreshStateMachine() – přegeneruje celý automat tj. všechny stavy a přechodové funkce

```
;(function ($, window, document, undefined) {

    $.widget('automat.base', {
        options: {
        _drawLine: function(canvasContext, radius, startCircle, endCircle, text, color ) {
        _getLinePoints: function (startPoint, endPoint, circle) {
        _qrt: function (number) {
        _drawPoint: function (p, name) {
        _drawArrow: function(canvasContext, p1, p2, color) {
        _drawCircle: function(canvasContext, circle, text, color) {
        _getStateIndex: function (name) {
        _drawCircleText: function(canvasContext, circle, text) {
        _refreshStateMachine: function () {
        _selectState: function (x, y) {
        _drawLineText: function (canvasContext, textPt, text) {
        _getPointOnCircle: function(radius, originPt, endPt) {
        _getAngleBetweenPoints: function (originPt, endPt) {
    })

})(jQuery, window, document);
```

Obrázek 5: Widget automat.base

Dalším ovladacím prvkem je automat.addAutomatGrafical, který rozšiřuje widget automat.base. V tomto widgetu jsou funkce, které jsou používány výhradně při tvorbě konečných automatů. Funkce jsou v následucích bodech popsány a také zobrazeny na obrázku 6.

- _addState() – přidává nový stav
- _editState() – upraví parametry vybraného stav, jako jsou například souřadnice, název, zda se jedná o hlavní nebo rozpoznávací stav
- _moveState() – funkce která pomocí táhnutí myši umožní přesunovat stav po kreslící ploše
- _removeState() – tato funkce odstraní vybraný stav a všechny jeho vstupní a výstupní přechodové funkce
- _addCrossing() – funkce na přidání přechodové funkce mezi dvěma stavy
- _generateStateTable() – vygeneruje přechodovou tabulku, včetně možností na úpravu a mazání stavů
- loadStateFromData() – slouží pro vygenerování stavů a přechodových funkcí z importu XML souboru

```
;(function ($, window, document, undefined) {

    $.widget('automat.addAutomatGrafical', $.automat.base, {
        options: {
        _create: function () {
        _mainFunctions: function () {
        _addState: function (event) {
        _editState: function (thisData) {
        _moveState: function () {
        _removeState: function (keyState) {
        _addCrossing: function (crossingStates, value) {
        _generateStateTable: function() {
        loadStateFomData: function (data) {
    });

})(jQuery, window, document);
```

Obrázek 6: Widget addAutomatGrafical

Posledním statickým ovládacím prvkem je automat.showAutomat, který opět rozšiřuje hlavní widget automat.base. Tento widget je zobrazen na obrázku 7 a slouží pro zobrazení konečného automatu. Má i příslušné funkce:

- _create() – funkce, která se pustí při inicializaci a slouží obecně pro simulaci
- _buildStateMachine() – ze všech zadaných stavů a přechodových funkcí vykreslí na canvas konečný automat
- _checkInputAplhabet() – funkce která před spuštěním konečného automatu překontroluje vstupní slovo a rozhodne, zda toto slovo automat příjme či nikoliv
- _showState() – po zadání stavu a písmene vstupní abecedy zobrazí stav a příslušnou výstupní přechodovou funkci
- _selectState() – po kliknutí v přechodové tabulce na stav ho v canvasu zobrazí včetně všech vstupních i výstupních přechodových funkcí

```
;(function ($, window, document, undefined) {

    $.widget('automat.showAutomat', $.automat.base, {
        options: {
        _create: function() {
        _buildStateMachine: function(statesData, statesCrossesData) {
        _checkInputAlphabet: function (inputAlphabet) {
        _showState: function (mode, startState, states) {
        _inArray: function (needle, haystack) {
        _selectState: function (input, line) {
        _unselectState: function (input) {
    });

})(jQuery, window, document);
```

Obrázek 7: Widget showAutomat

Další a poslední část v API je již dynamická. Ta se generuje z databáze podle daného typu konečného automatu. Z této databáze se vytváří dva nové widgety, které rozšiřují addAutomatGrafical a showAutomat. Jsou určené k tomu aby si administrátor mohl přetížit funkce _addCrossing(), _addState(), _editState() a _removeState() ve widgetu addAutomatGrafical. Ve widget showAutomat se dají přetížit funkce _checkInputAplhabet() a _create().

4.4.2. Práce s API

Nyní si řekneme, jak vytvoříme například vytvořit nový typ automatu. Výchozí stav API je připraven na vytvoření nedeterministického automatu, to znamená, že jeden stav může mít několik přechodových funkcí se stejnou hodnotou.

V případě vytvoření deterministického typu automatu, můžeme postupovat následujícím způsobem:

1) Přihlásíme se do administrace pod svým uživatelským účtem
2) V horním horizontálním menu vybereme položku „Typy automatů“
3) Zvolíme možnost „Nový typ automatu“
4) Chceme-li upravovat chování při zadávání přechodové funkce, najdeme v záložce „Zadávání automatu“ pole „Přechodová funkce“
5) Zde přidáme kontrolu na výstupní přechodový stav, tak jak je znázorněno na následujícím obrázku 8

```
_addCrossing: function (crossingStates, value) {
    var sameState = false;
  $.each(self.options.statesCrossing, function(k, v) {
      if (v.startState.text == self.options.states[crossingStates.start].text && v.value == value) {
          sameState = true;
        }
    });
    if (sameState === true) {
      alert("Z tohoto stavu již přechodová funkce s hodnotou '"+value+"' exustuje");
    } else {
      self.options.statesCrossing[self.options.statesCrossing.length] = { // přidáme přechodový stav do seznamu stavů
          startState: self.options.states[crossingStates.start],
          endState: self.options.states[crossingStates.end],
            value: value
        };
    }
},
```

Obrázek 8: Úprava API na deterministický automat

6) Zde naše práce končí a po uložení již můžeme tento nový typ automatu začít používat. Pokud jsme tedy vše nastavili správně, tak by automat neměl umožnit vytvořit více přechodových funkcí z jednoho stavu se stejnou hodnotou

Tímto způsobem lze upravit jakékoliv chování automatu, jednak při zadávání tak při jeho simulaci. Nemusíme upravovat fyzicky kód, například přes FTP.

4.5. Hosting aplikace

Pro správný běh aplikace je nutno projekt umístit na webhosting nebo na virtuální server. V našem případě jsem zvolil umístění na vlastní virtuální server. Z tohoto důvodu máme všechna data pod kontrolou a jakékoliv nastavení si můžeme kdykoliv upravit podle našich požadavků

Na serveru jsem vytvořil virtuální stroj a doménu ondrajodas.eu nasměroval právě na tento server. Pro obsluhu souborů projektu používám verzovací program GIT, pomocí kterého nahrávám veškeré soubory na server. Správu databáze zajišťuje správce databází, známý pod názvem phpMyAdmin.

5. Popis programu

Internetové stránky jsou zveřejněné na adrese http://www.ondrajodas.eu/ a mají intuitivní grafické rozhraní, pomocí kterého lze jednoduše vytvářet a následně simulovat konečné automaty.

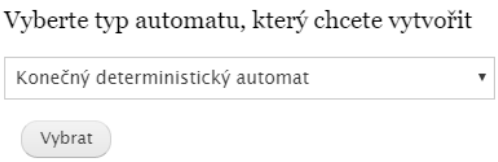
V této kapitole si popíšeme, jakým způsobem se provádí zadávání nových konečných automatů, simulace zadaných automatů, export a import automatů pomocí XML a vytváření nových typů automatů pomocí API.

5.1. Úvodní stránka programu

Obrázek 9: Hlavní stránka programu

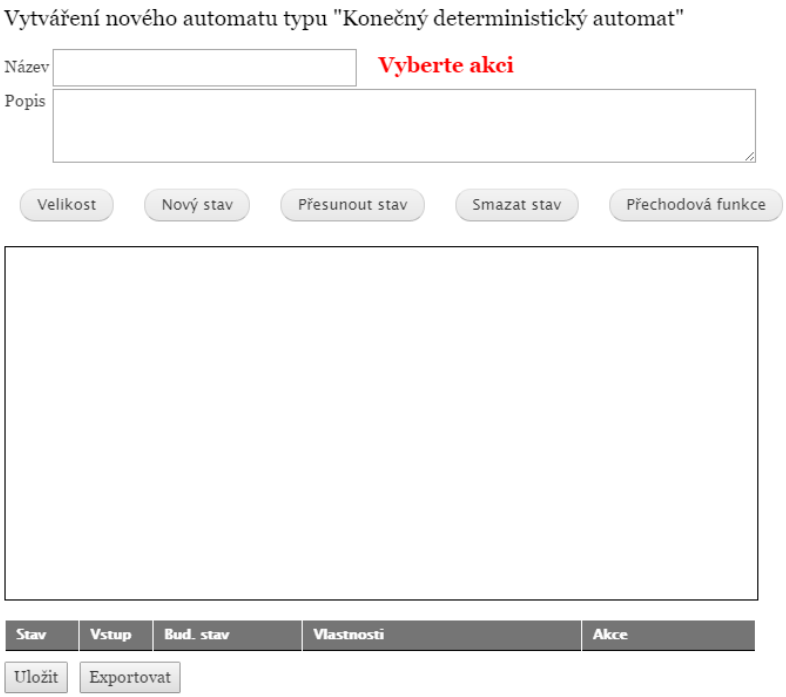
Na úvodní stránce programu, která je zobrazená na Obrázek 9: Hlavní stránka programu nalezneme veškeré zadané automaty, možnost vytvořit automat nový, nebo naimportovat z XML. Dále je možné jednotlivé automaty simulovat nebo smazat. Všechny tyto funkce si popíšeme v následujících kapitolách.

5.2. Zadávání nového konečného automatu



Obrázek 10: Výběr typu automatu

Před samotným zadáváním automatu si nejprve vybereme, jaký typ chceme vytvářet. Výběr typu je zobrazen na obrázku 10. Vytváření nových typů je provedeno pomocí javascriptového API a popíšeme si ho v dalších kapitolách.



Obrázek 11: Výchozí stav zadávání automatu

Výchozí stav při zadávání automatu je zobrazen na obrázku 11. Hlavní část je vytvořena technologií HTML5 canvas, která slouží ke grafickému zobrazení programu. Nad kreslící plochou jsou umístěny hlavní funkce, které lze zadávat. Nyní následuje popis těchto funkcí:

- Velikost - po výběru se nám zobrazí dialogové okno pro změnu velikosti kreslící plochy konečného automatu

- Nový stav - po výběru se nám otevře dialogové okno, kde si můžeme zadat vlastnosti nového stavu. Poté co si stav nadefinujeme kliknutím myší na kreslící plochu, umístíme stav do programu. Rozlišujeme tři druhy nového automatu. Prvním je normální stav, který nemá žádnou speciální vlastnost 

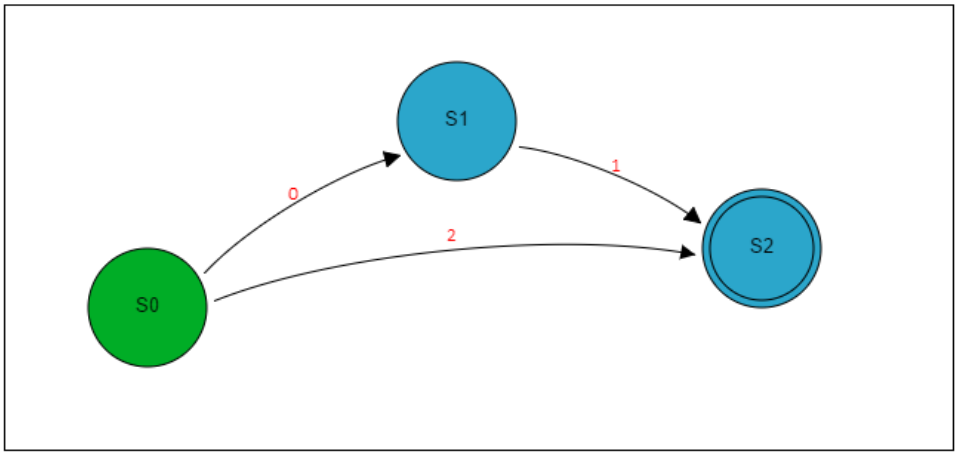
. Pak počáteční stav, který se zobrazuje 

, z tohoto stavu program začíná s průchodem automatu. A posledním typem stavu je rozpoznávací stav, který označujeme 

.
- Přesunout stav - tato funkce nám po kliknutí a tažením na existující stav změní polohu stavu konečného automatu
- Smazat stav - po výběru a kliknutí na jeden ze stavů nám ho smaže i se všemi vstupními a výstupními přechodovými funkcemi
- Přechodová funkce - po výběru této funkce se nám zobrazí dialogové okno, ve kterém zadáme hodnotu přechodové funkce a poté vybereme počáteční a koncový stav a tím vytvoříme mezi nimi propojení

Ze všech těchto funkcí můžeme poskládat libovolný konečný automat. Například jako na obrázku 12. Další částí při zadávání konečného automatu je přechodová tabulka umístěna pod grafickou částí, doplněná o vlastnosti stavů a možnost stav smazat nebo upravit.

Tato tabulka zobrazuje všechny stavy a jejich vlastnosti a veškeré přechodové funkce mezi stavy. Z této tabulky je možno kterýkoliv stav změnit nebo ho smazat.



Stav	Vstup	Bud. stav	Vlastnosti	Akce
S0	0	S1	x:96, y:204, POČÁTEČNÍ STAV	Upravit Smazat
	2	S2		
S1	1	S2	x:305, y:78	Upravit Smazat
S2			x:511, y:164, rozpoznávací	Upravit Smazat

Uložit Exportovat

Obrázek 12: Zadávání konečného automatu

Poslední částí je uložení a vyexportování automatu. Uložení, jak sám už název napovídá, představuje uložení celého automatu do databáze. V tu chvíli již nelze s automatem dále pracovat. Tedy je již v neměnném stavu.

Proto je zde tlačítko exportovat, které nám automat vyexportuje ve formátu XML. Pokud bychom chtěli dělat další nějaké úpravy, tak lze automat zpětně naimportovat z tohoto souboru. Tento způsob jsem zvolil, protože vytvoření a simulace automatů probíhá bez jakéhokoliv přihlášení do systému. Není tedy žádoucí, aby uživatelé měnili hotové automaty, ale pouze je vytvářili, popřípadě simulovali.

5.3. Import a export konečného automatu pomocí XML

Import a export konečného automatu se provádí pomocí XML souboru, který má přesnou strukturu, viz obrázek 13.

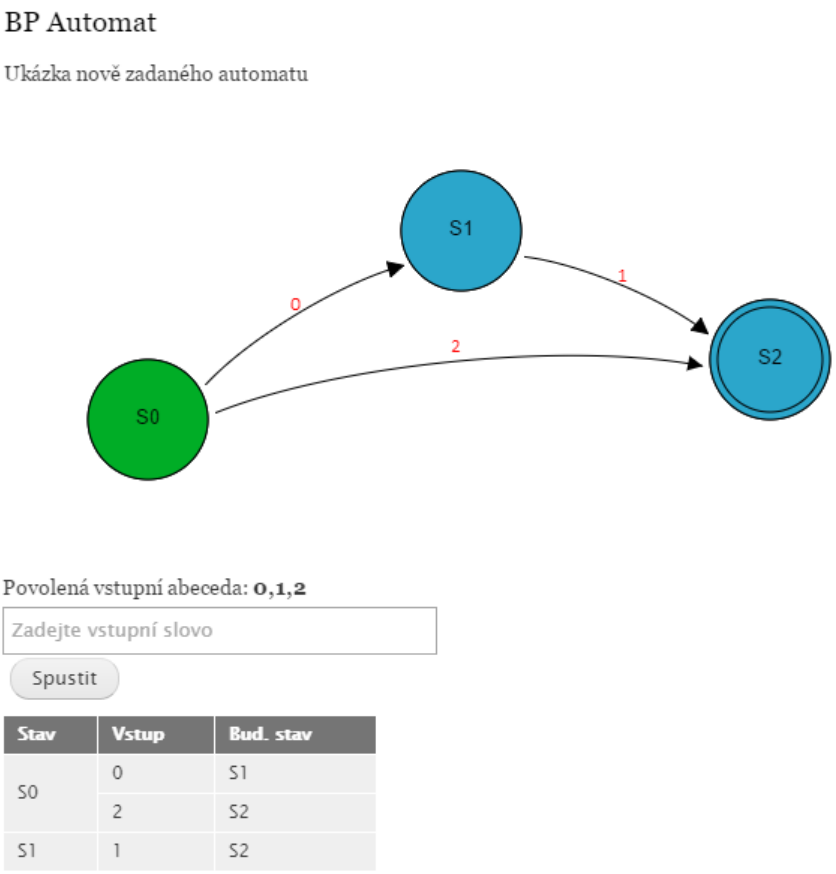
Import můžeme provést ze stránky se seznamem automatů. Po úspěšném naimportování se nám zobrazí stránka pro zadávání nového konečného automatu. Rozdíl je v tom, že jsou již zadány stavy a přechody, které byly nadefinovány v XML.

Vyexportovat si můžeme již rozpracovaný automat ale i automat, který je hotový. Posléze můžeme zpětně naimportovat a provést patřičné změny. Tím se nám uloží automat nový a do původního nebudeme nijak zasahovat.

```
▼<export>
  ▼<fa type="1" width="640" height="300">
    ▼<name>
       <![CDATA[ bp ]]>
     </name>
    ▼<description>
       <![CDATA[ ukázkový automat bakalářské práce ]]>
     </description>
    ▼<states>
      ▼<state>
         <name>S0</name>
         <position x="102" y="190"/>
         <recognition>0</recognition>
         <main>1</main>
       </state>
      ▼<state>
         <name>S1</name>
         <position x="304" y="58"/>
         <recognition>0</recognition>
         <main>0</main>
       </state>
      ▼<state>
         <name>S2</name>
         <position x="509" y="145"/>
         <recognition>1</recognition>
         <main>0</main>
       </state>
     </states>
    ▼<states_crossing>
       <state_crossing from_state="S0" to_state="S1" value="0"/>
       <state_crossing from_state="S1" to_state="S2" value="1"/>
       <state_crossing from_state="S0" to_state="S2" value="2"/>
     </states_crossing>
   </fa>
</export>
```

Obrázek 13: XML struktura automatu

5.4. Simulace konečného automatu



Obrázek 14: Simulace automatu

Mnou vytvořený automat, který jsem v předchozích kapitolách představil je připraven pro simulaci. Základní obrazovka je zobrazena na obrázku 14. Mám zde graficky zobrazený automat, input na zadání vstupního slova pro simulaci a přechodovou tabulku.

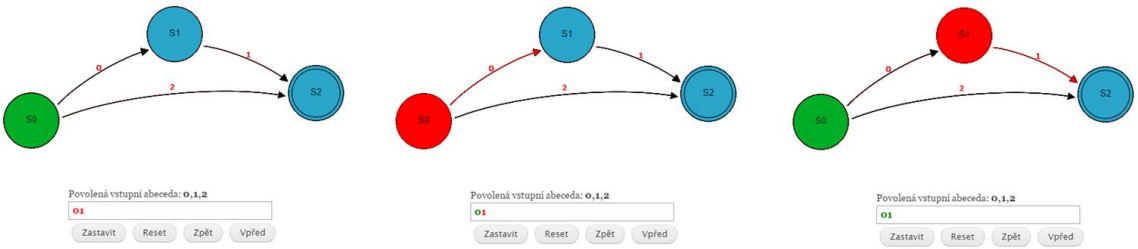
Po zadání vstupního slova a kliknutí na tlačítko spustit, nám automat vyhodnotí, zda slovo příjme či nikoliv. Pokud dojde k zamítnutí, vyskočí upozornění, že slovo nebylo přijato. Pokud ale dojde k přijetí slova, objeví se nám tlačítka pro simulaci, která jsou zobrazena na obrázku 7.

Obrázek 15: Rozhraní simulace automatu

Objevily se nám čtyři funkční tlačítka, kterými jsou:

- Zastavit – toto tlačítko nám zastaví simulaci a vrátí nás do předchozího stavu pro zadání vstupního slova
- Reset – vynuluje nám simulaci a vrátí nás do počátečního stavu simulace
- Zpět – posune simulaci o jeden předchozí vstupní znak ve slově
- Vpřed – posune simulaci o jeden následující vstupní znak ve slově

Pokud tedy spustíme simulaci a budeme klikat na tlačítko vpřed, dojdeme do stavu jako je na obrázku 16. Stav, ve kterém se právě nachází automat, je značen červeně. Aktivní přechodové funkce jsou zobrazeny taktéž červenou barvou. Dále vstupní slovo je zelenou barvou obarvena ta část, která už simulací prošla. Červenou barvou naopak část, která ještě nikoliv.

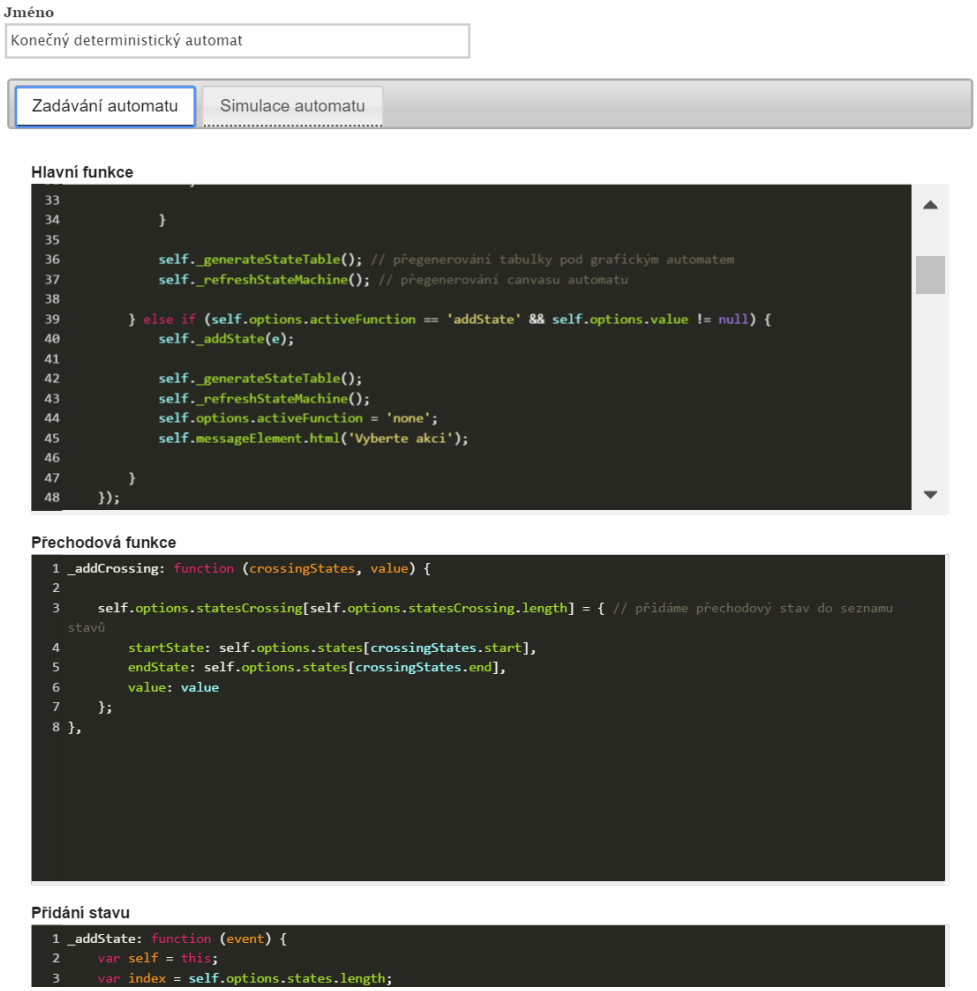


Obrázek 16: Průběh simulace automatu

5.5. Zadávání nových typů automatů pomocí API

Funkce pro zadávání nových typů je přístupná pouze po zaregistrování a schválení administrátorem webových stránek. Umožňuje upravit veškeré chování konečných automatů. Můžeme si zde například nadefinovat, jak se bude automat chovat po přidání nového stavu nebo po propojení stavů přechodovou funkcí. Lze i upravovat stávající typy, které jsou již nadefinované.

Obrazovka pro zadávání nového typu je na obrázku 17.



Obrázek 17: Obrazovka pro zadávání nového typu konečného automatu

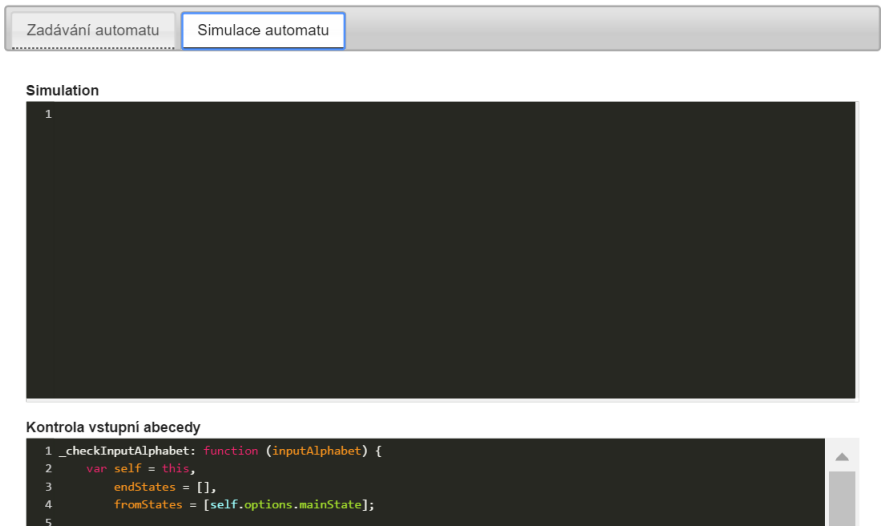
Pro ulehčení zadávání jsou nadefinované rychlé klávesy, které jsou v následujících bodech popsány:

- F9 – pro rychlé uložení typu automatu (dostupné pouze při editaci)
- F11 – zobrazení na celou obrazovku aktuálně upravovaného kódu
- Ctrl+mezerník – našeptávač javascriptového kódu

Dále jsou zde dostupné předdefinované funkce a proměnné:

- Options.state – seznam všech stavů, které vykreslí na kreslící plochu
- Options.stateCrossing – seznam přechodů mezi stavy
- Options.mainState – určení výchozího stavu, ze kterého bude procházení začínat
- Self.refreshStateMachine() – funkce která překreslí celý automat (např. po přidání nového stavu)
- Self.generateStateTable() – funkce pro vygenerování přechodové tabulky

Zadávání funguje na základě přetěžování metod javascriptového API. To znamená, že pokud některou metodu chceme zachovat v původním stavu, nemusíme jí opisovat z defaultního automatu, ale stačí ponechat pole prázdné jako je tomu v sekci „Simulace automatu“ na obrázku 18.



Obrázek 18: Dědění metod

6. Závěr

V první části mé bakalářské práce jsem čtenáře seznámil s konečný automatem. Zaměřil jsem se zejména na způsob, jak prochází automat vstupní abecedu a popis významu všech použitých pojmů. Dále z čeho se automat skládá a jaké jsou jeho prvky.

V následující kapitole jsem provedl rešerši nejvíce používaných redakčních systémů, které jsem následně porovnával. Sledoval jsem důležité prvky a následně zvolil vhodný redakční systém pro můj projekt. Vybraným redakčním systémem se stal Drupal. Na tomto systému jsem vytvořil interaktivní moderní webovou aplikaci, kterou čtenář nalezne na adrese http://www.ondrajodas.eu/. V této aplikaci je možné vytvářet a simulovat konečné automaty. Pro registrované uživatele nabízí aplikace také možnost upravovat API pro nové typy automatů. Vše jsem připravil a naprogramoval jako pomůcku pro výuku teoretické informatiky. Proto si myslím, že nejvíce čtenářů mé bakalářské práce a zároveň uživatelů mnou vytvořených webových stránek bude z řad studentů fakulty Mechatroniky. Stránky jsou plně funkční a k dispozici studentům.

Další část této práce obsahuje popis API a jeho možnosti při programování nového typu automatu.

V poslední části se může čtenář seznámit se samotnými webovými stránkami, kde nalezne i názornou ukázku příkladu vytvoření konečného automatu a jeho simulaci.

V navazujícím studiu a zřejmě námětem pro mou diplomovou práci bude rozšíření webových stránek pomocí regulárních jazyků.

Internetové zdroje

[1].*Jančar, Petr*. Teoretická informatika. Dostupné z http://www.cs.vsb.cz/sawa/uti/materialy/ti.pdf [cit. 15.2.2016]

[2] *W3Techs*. Usage of content management systems for websites [online]. Dostupné z WWW: http://w3techs.com/technologies/overview/content_management/all [cit. 21.11.2015]

[3] *Alexa*. Načteno z Alexa: http://www.alexa.com/ [cit. 21.11.2015]

Seznam literatury

[4] *Cecco, Raffaele*. JavaScript Graphics. First edition. O'Reilly media, july 2011. 260 s. ISBN: 978-1-449-39363-2

[5]*Fulton, Steve; Fulton, Jeff*. HTML5 Canvas. First edition. O'Reilly media, may 2011. 628 s. ISBN: 978-1-449-39390-8

Seznam příloh

Příloha A - Použité technologie a knihovny

Jak jsem zmínil již v rešerši, zvolil jsem takové technologie, aby internetové stránky byly dostupné na webu bez jakékoliv instalace a zároveň které odpovídají osobním preferencím. Z toho důvodu jsem vyloučil Javu nebo Flash a jím podobné technologie pro které je třeba instalovat dodatečný software. Pro snadnější práci s javascriptem jsem zvolil použití knihovny jQuery, jQuery UI a pro grafickou část HTML5 canvas.

V další části kapitoly se budu zabývat detailnějším popsáním použitých technologií a architekturou aplikace.

PHP

PHP[1] je zkratka hypertextového prepocesoru a slouží k dynamické tvorbě internetových stránek a webových aplikací ve formátu HTML nebo XHTML. Pro jeho běh není na straně klienta nutno instalovat žádný speciální software, postačí mu pouze internetový prohlížeč.

Při jeho použití, je na straně serveru, spuštěn požadovaný skript jeho výsledek je poté odeslán do prohlížeče klienta, který ho vykreslí jako statickou HTML stránku.

MySQL

MySQL[2] je multiplatformní databázový systém, který je dostupný pod licencí GPL, proto je jednou z nejrozšířenější databází pro tvorbu webových stránek.

jQuery

jQuery[3] je javacriptová knihovna, která pracuje s elementy DOM. Místo dlouhých názvů, jako například document.getElementById(´idElementu´) se používá zkráceny jQuery tvar $(´idElementu´). Obdobně se používám výběr podle třídy nebo typu elementu stejně jak je tomu v CSS.

Po výběru určitého elementu lze jednoduše měnit jeho atributy jako například CSS třídy nebo data atributy apod.

---

[1] http://php.net/
[2] https://www.mysql.com/
[3] https://jquery.com/

jQuery UI

jQuery UI[4] je postaveno na knihovně jQuery a je zaměřen na uživatelské rozhraní. Dělí se do čtyř základních částí, kterými jsou:

- Core – jádro knihovny které je potřeba ke všem součástem
- Interactions – obsahuje metody na interakci mezi uživatelem a aplikací jako jsou například draggable, droppable atd.
- Widgets – obsahuje elementy UI pro vyskakovací okna, tlačítka a progressbary. Jako příklad si můžeme uvést dialog, datepicker, autocomplete nebo slider.

jQuery UI je podporováno všemi velkými internetovými prohlížeči a je vyvíjeno pod licencí MIT a GPL.

HTML5 canvas

Canvas[5] je v html speciální element pro grafické kreslení na webových stránkách. Kreslení provádíme skriptováním, nejčastěji pomocí jazyka javascript.

Codemirror

Codemirror je javascriptový textový editor pro psaní javascriptu nebo kaskádových stylů. Umožňuje editaci kódu a zvýrazňuje syntaxi, jak je vyobrazeno na obrázku 19. Je aplikováný pro úpravu javascriptového API.

```
1  _mainFunctions: function () {
2      var self = this,
3          crossingStates = {
4              start: false,
5              end: false
6          };
7
8      self.element.mousedown(function (e) {
9
10         var keyState = self._selectState(e.offsetX, e.offsetY); // zjistime jestli jsme klikli na existujici
   stav
11
12         if (keyState !== false) { // klikli jsme na stav
13             if (self.options.activeFunction == 'removeState') { // chceme vymazat stav
14
15                 self._removeState(keyState);
```

Obrázek 19: Zvýraznění syntaxe pomocí codemirror

[4]https://jqueryui.com/
[5]http://www.w3schools.com/html/html5_canvas.asp